Draft Recommendation ITU-T X.itssec-1

Secure software update capability for intelligent transportation system communication devices

Summary

As intelligent transportation system (ITS) technologies improve, it has become common for vehicles to communicate with other entities such as other vehicles, vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications. Since electric devices inside a vehicle such as electronic control units (ECUs), and electric toll collections (ETCs), system and car navigation systems are becoming more sophisticated. As a result, software modules inside those electric devices need to be appropriately updated for the purpose of bug fixing, and for performance and security improvements to avoid crucial accidents.

In order to fulfil the above requirement, Recommendation ITU-T X.itssec-1 provides secure software update procedures with appropriate security controls based on threat analysis. This Recommendation can be practically utilized by car manufactures and ITS-related industries as a set of standard capabilities for best practices.

Keywords

Communication devices, denial of service (DoS) attack, embedded system, hardware security module (HSM), intelligent transportation system (ITS), malware, privacy, risk analysis, vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), vehicle-to-X (vehicle/infrastructure) (V2X), wireless communications.

Contact:	Masashi Eto NICT Japan	Tel: +81 42 327 5573 Fax: +81 42 427 6640 Email: eto@nict.go.jp
Contact:	Koji Nakao NICT / KDDI Japan	Tel: +81 42 327 5821 Fax: +81 42 427 6640 Email: ko-nakao@nict.go.jp

CONTENTS

1 Scope 5

2 References 5

3 Definitions 5

3.1 Terms defined elsewhere 5

3.2 Terms defined in this Recommendation 5

4 Abbreviations and acronyms 6

5 Conventions 7

6 Basic model of remote software update 8

6.1 Modules of ITS environment for software update 8

6.1.1 User interface 8

6.1.2 Electronic control unit (ECU) 8

6.1.3 Vehicle mobile gateway 8

6.1.4 Update server and log database 9

6.1.5 Supplier 9

6.2 Model of software update procedure 9

7 Threats and risk analysis 11

7.1 Definition of target system of evaluation 11

7.2 Identification of threats with higher risks 13

7.2.1 TOE security requirements 18

7.2.1.1 SR.integrity/availability protection of VMG functions through CAN communication 18

7.2.1.2 SR.confidentiality protection of VMG data 19

7.2.1.3 SR.integrity/availability protection of VMG functions through mobile communication 19

7.2.1.4 SR.FaultTolerance of VMG functions 19

7.2.1.5 SR.integrity/availability protection of VMG functions through OBD 19

7.2.1.6 SR.confidentiality/integrity/availability protection of VMG through Wi-Fi communication 19

7.2.2 Security requirements for the operational environment of TOE from IT perspective 19

7.2.2.1 SRE.ECU protection 19

7.2.2.2 SRE.CAN communication protection 20

7.2.2.3 SRE.Mobile communication network protection 20
7.2.2.4 SRE.Wireless communication protection 20
7.2.3 Security requirements for the operation environment from Non-IT operation/management perspective 20
7.2.3.1 SREN.Caution 20
7.2.3.2 SREN.NetworkServicer 20
7.2.3.3 SREN.OBD tool protection 20
7.2.3.4 SREN.User 20
7.2.3.5 SREN.VirusScan 20
7.2.3.6 SREN.Wireless-Device protection 20
7.2.3.7 SREN.Wireless-Display 21
7.3 Security controls 21
7.3.1 SC.Trusted boot 21
7.3.2 SC.Message verification 21
7.3.3 SC.Authentication of communication entity 21
7.3.4 SC.Message filtering 22
7.3.5 SC.FaultTolerance of VMG functions 22

8 Specification of secure software update procedure 22
8.1 General message format with security functions 23
8.1.1 Digital signature method 23
8.1.2 MAC method 23
8.2 Protocol definition and data format 23
8.2.1 Protocol overview 23
8.2.2 diagnose messages 25
8.2.2.1 diagnose (request) message 25
8.2.2.2 diagnose (report) message 26
8.2.2.3 diagnose (submit) message 27
8.2.2.4 diagnose (receipt) message 29
8.2.3 update_check messages 31
8.2.3.1 update_check (request) message 31
8.2.3.2 update_check (response) message 33
8.2.4 update messages 35
8.2.4.1 update (notification) message 36
8.2.4.2 update (confirmation) message 38
8.2.4.3 update (application) message 39
8.2.4.4 update (result) message 41
8.2.5 update_report messages 43
8.2.5.1 update_report (submit) message 43
8.2.5.2 update_report (receipt) message 45

Appendix I..... 48

I.1 Methodology on risk analysis based on [b-JASO TP15002]..... 48

I.1.1 Phase 1: Definition of the target of evaluation..... 48

I.1.2 Phase 2: Identification of threats..... 51

I.1.3 Phase 3: Risk analysis..... 52

I.1.3.1 CRSS..... 52

I.2 Data verification using MAC algorithms..... 55

Bibliography..... 56

1 Scope

In the context of updates of software modules in the electric devices of vehicles in the intelligent transportation system (ITS) communication environment, this Recommendation aims to provide a procedure of secure software updating for ITS communication devices for the application layer in order to prevent threats such as tampering of and malicious intrusion to communication devices on vehicles. This includes a basic model of software update, its threat and risk analysis, security requirements and controls for software update and a specification of abstract data format of update software module.

The procedure is intended to be applied to communication devices on ITS vehicles under vehicle-to-infrastructure (V2I) communication by means of the Internet and/or ITS dedicated networks. The procedure can be practically utilized by car manufactures and ITS-related industries as a set of standard secure procedures and security controls.

2 References

The following ITU-T Recommendations and other references contain provisions which, through reference in this text, constitute provisions of this Recommendation. At the time of publication, the editions indicated were valid. All Recommendations and other references are subject to revision; users of this Recommendation are therefore encouraged to investigate the possibility of applying the most recent edition of the Recommendations and other references listed below. A list of the currently valid ITU-T Recommendations is regularly published. The reference to a document within this Recommendation does not give it, as a stand-alone document, the status of a Recommendation.

ITU-T X.509 Recommendation ITU-T X.509 (2012) | ISO/IEC 9594-8:2014, *Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks*.

ITU-T X.1521 Recommendation ITU-T X.1521 (2011), *Common vulnerability scoring system*.

ISO/IEC 15408-1:2009 *Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model*.

ISO/IEC 27000:2014 *Information technology -- Security techniques -- Information security management systems -- Overview and vocabulary*

3 Definitions

3.1 Terms defined elsewhere

This Recommendation uses the following term defined elsewhere:

3.1.1 threat [ISO/IEC 27000:2014]: potential cause of an unwanted incident, which may result in harm to a system or organization.

3.2 Terms defined in this Recommendation

This Recommendation defines the following terms:

3.2.1 risk score: A score that a risk analysis method calculates for each threat.

3.2.2 vehicle mobile gateway (VMG): A module which provides communication between electronic control units (ECUs) in the controller area network (CAN) (in-vehicle buses) and exterior intelligent transportation system (ITS) entities in the external network.

4 Abbreviations and acronyms

This Recommendation uses the following abbreviations and acronyms:

CA	Certification Authority
CAN	Controller Area Network
CD	Compact Disc
CRSS	CVSS based Risk Scoring System
CVSS	Common Vulnerability Scoring System
DoS	Denial of Service
DVD	Digital Versatile Disc
ECU	Electronic Control Unit
ETC	Electronic Toll Collection
FT	Fault Tree
GPS	Global Positioning System
GUID	Global User ID
HSM	Hardware Security Module
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ID	IDentifier
IT	Information Technology
ITS	Intelligent Transportation System
LIN	Local Interconnect Network
MAC	Message Authentication Code
MOST	Media Oriented Systems Transport
OBD	On-board diagnostics
OEM	Original Equipment Manufacturer
PC	Personal Computer
RPM	Revolutions per Minute
RSS	Risk Scoring System
SD	Secure Digital
SHA	Secure Hash Algorithm
SHE	Secure Hardware Extension
SSL	Secure Socket Layer
TLS	Transport Layer Security
TOE	Target of Evaluation
TPM	Trusted Platform Module

TV	TeleVision
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
Usvr	Update server
V2D	Vehicle-to-mobile-Device
V2I	Vehicle-to-Infrastructure
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-X (vehicle/infrastructure)
VMG	Vehicle Mobile Gateway
Wi-Fi	Wireless-Fidelity
XML	eXtended Markup Language

5 Conventions

None.

6 Basic model of remote software update

For considerations of a practical security architecture, this clause introduces a basic model of conventional architecture of software update, where a definition of principal modules and typical software update processes are provided.

6.1 Modules of ITS environment for software update

Figure 1 provides a view of principal modules around a vehicle for a remote software update in the ITS communication environment. The principal modules are information devices, electronic control units (ECUs), and vehicle mobile gateway on a vehicle, update server and log database of car manufacturer and supplier.

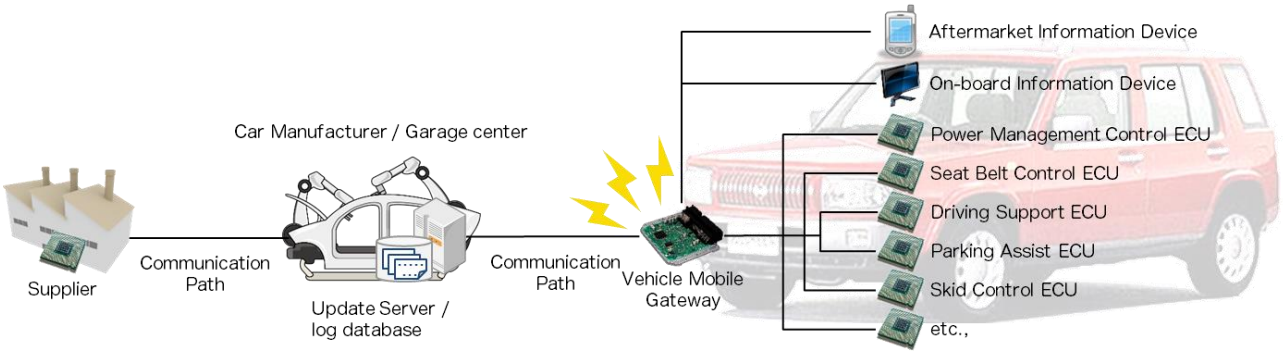

Figure 1 – Principal modules around a vehicle

6.1.1 User interface

The user interface is generally an on-board or an aftermarket information device with display and input devices on a vehicle. Such an information device is directly connected to other devices on a vehicle (e.g. vehicle mobile gateway (VMG) or ECU (see clause 6.1.2)) so that it can obtain and indicate various status information of the vehicle such as speed, revolutions per minute (RPM), fuel level, and so on. Particularly, in this Recommendation, a user interface is used to notify car drivers of the necessity for updates.

6.1.2 Electronic control unit (ECU)

ECU is a generic term for computers which control various kinds of devices in a vehicle. In the earliest years of ECU, its main functions were controls of ignition timing, injection, idling adjustment and limiter of engine in order to improve fuel efficiency and reduce gas emissions. According to the computerization of the vehicle, ECU has expanded its applications to various kinds of functions such as power management, seat belt control, driving support, parking assist, skid control, automatic transmission, and so on. In recent years, the number of ECUs within a vehicle increased from 50 to 100, and the importance of ECUs for safety control and communication is especially growing. However, since the development of ECU involves sophisticated software implementations, the recent increase of ECUs in vehicles imposes a heavy burden on car manufacturers.

6.1.3 Vehicle mobile gateway

A vehicle mobile gateway (VMG) is a module assigned for the software-update management of ECUs and connection management between ECUs in controller area networks (CANs) (in-vehicle buses) and exterior ITS entities. VMG is a conceptual entity which can be practically implemented with a set of multiple components. For example, the connection management entity (a.k.a. "central gateway", "Head unit" or "communication head unit") can be used for the role of VMG in this context, and any devices can be also used for the software-update management of ECUs. A cellular

network (mobile network) and a fixed network via wireless are applied as a communication path between vehicle mobile gateway and exterior ITS entities.

6.1.4 Update server and log database

An update server is located at car manufacturers or garage centres to gather status information of software modules from vehicles and to distribute software update modules to the vehicles. Likewise, in most recent networked computers such as personal computers (PCs) and smart phones, one of the important functions of the update server is to entirely manage and control software on vehicles. In order to automatically manage such software status of each vehicle, the update server should work with a log database which stores such software status information of the vehicle as evidence. Note that an update server can be deployed not only at a car manufacturer but also at a supplier or third party.

6.1.5 Supplier

A vehicle is an assembly of thousands of automotive parts that are provided by automotive suppliers. On-board communication devices and ECUs are provided by those suppliers and assembled by car manufacturers while considering dependencies among the various devices. Therefore, and in general, update modules for on-board communication devices are firstly produced not by a car manufacturer but by a corresponding supplier. The provided update modules are distributed from a car manufacturer to vehicles after careful testing and evaluation at the car manufacturer.

6.2 Model of software update procedure

Figure 2 shows a typical model of software update procedure which is initiated by the vehicle mobile gateway by checking the existence of updates.

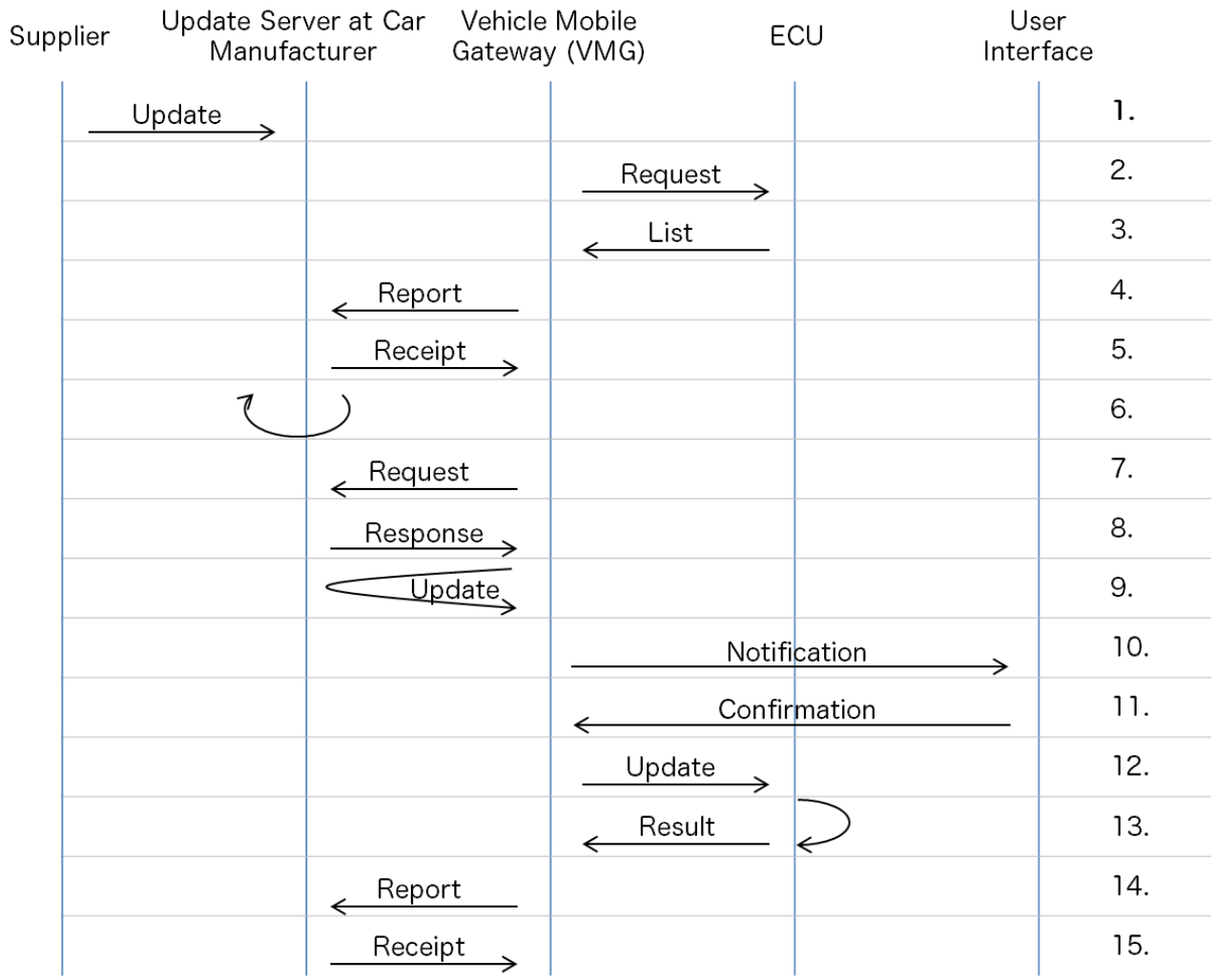

Figure 2 – Model of software update process

The update procedure is as follows:

1. At the first step of the process, an update module is provided by an automotive component supplier, which occurs asynchronously with the following steps.
2. As the initiation of the update procedure starts, a vehicle mobile gateway (VMG) requests ECUs to submit their software list.
3. An ECU diagnoses itself, generates a list of software modules and reports it to VMG.
4. VMG submits the collected list to the update server to check whether any update for the vehicle exists.
5. The update server sends back a receipt of the submitted list to VMG.
6. According to the list, the update server inspects the status of the installed software of the vehicle and determines the necessary software updates for the ECUs.
7. Since this inspection may take a long time, VMG periodically checks the necessity of the updates for the vehicle.
8. If there is any update, the update server sends an access uniform resource locators (URLs) for the updates; otherwise, it sends back only an acknowledgement message.

9. If there is any update for the vehicle, VMG connects to the update server to download the update modules for the vehicle.
10. Before applying the updates to ECUs, VMG notifies the driver to confirm the application of the updates.
11. The driver confirms and accepts to apply the updates.
12. VMG delivers the update files to the corresponding ECUs and requests them to apply the updates.
13. Each ECU applies the update and reports the application result to the vehicle mobile gateway.
14. The vehicle mobile gateway submits a report of application results to the update server.
15. Finally the update server sends back a receipt of the update information. If the application of the update has failed or some remaining update is found, the update server retries the procedure from step 6 to 14 until the application has succeeded.

7 Threats and risk analysis

Examples of the various security attacks are known to the information technology (IT) system so far, and the know-how to evaluate a risk is accumulated by the IT system design. The basic security concepts necessary for evaluation of IT products are given in [ISO/IEC 15408-1]. In the context of evaluation, [ISO/IEC 15408-1] uses the term target of evaluation (TOE). There are assets which are entities that the owner of TOE presumably places value upon. [ISO/IEC 15408-1] aims to establish security objectives for TOE, which is a statement of an intent to counter identified threats and/or satisfy identified organization security policies and/or assumptions. Threats give rise to risks to the assets, based on the likelihood of a threat being realized and the impact on the assets when that threat is realized. However, [ISO/IEC 15408-1] does not specify how to perform threat extraction and risk analysis. On the other hand, there are a number of known methods for threat extraction and risk analysis. In this clause, taking one of these methods, based on the guideline concerning automotive information security [b-JASO TP15002], threat identification and risk analysis are performed. This guideline is used in an informative manner. The detailed information on it is provided in Appendix I.

7.1 Definition of target system of evaluation

This clause defines target of evaluation (TOE).

TOE is defined as VMG. As an interface to the outside, there are an on-board diagnostics (OBD) connector, a mobile communication module, a global positioning system/global navigation satellite system (GPS/GLONASS) signal reception device, a wireless fidelity (Wi-Fi), a radio/television (TV), Bluetooth connection, a CAN0/1 connection, a user interface with a compact disc/digital versatile disc (CD/DVD), a universal serial bus (USB) connector, and a secure digital (SD) connector. Though the controller area connector (CAN) is applied as one of the in-vehicle buses in this clause, identical analyses can be applied to the other types of in-vehicle buses such as Media Oriented Systems Transport (MOST), Local Interconnect Network (LIN) and FlexRay, etc.

In Figure 3, target of evaluation (TOE) is the area surrounded by the dotted line and it realizes secure communication control as a connection interface to the outside of the vehicle.

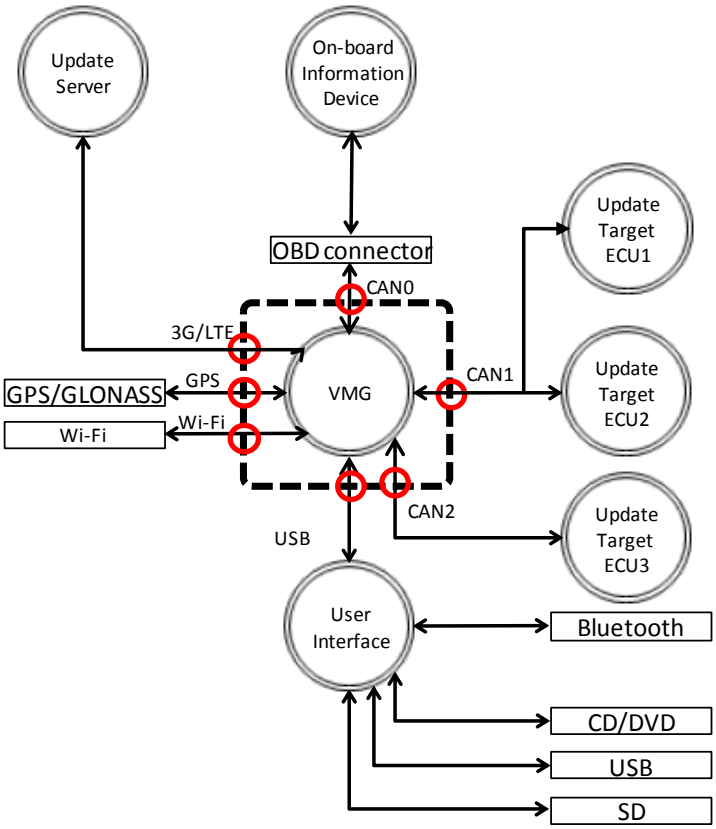

Figure 3 – TOE model

The overview of functions of the modules in TOE are shown in Table 1.

Table 1 – Overview of TOE module functions
(The format is referred to [b-JASO TP15002] Table B.1)

#	Module	Function		Asset	C	I	A
1	Vehicle mobile gateway	Mobile communication function	It communicates with the server via mobile connection. It uses authentication information to authenticate the server.	Mobile communication function		Y	Y
				Authentication information	Y	Y	
		Software get function	It gets software remotely via mobile connection or via OBD connector.	Software get function		Y	Y
				Software information	Y	Y	
		Remote software update function	It updates software remotely via mobile connection or via OBD connector. If software is updated remotely, it uses security information for update to authenticate the server.	Remote software update function		Y	Y
				Security information for update	Y	Y	
				Software information	Y	Y	

#	Module	Function		Asset	C	I	A
		GPS reception function	It receives data from GPS satellite.	GPS reception function		Y	Y
		Wi-Fi connection function	It establishes Internet connection of devices via Wi-Fi connection. It uses authentication information via Wi-Fi connection	Wi-Fi connection function		Y	Y
				Authentication information	Y	Y	
		USB connection function	It communicates via USB cable with the user interface.	USB connection function		Y	Y
		CAN communication function	It sends/receives CAN data to/from ECU	CAN communication function		Y	Y
		CAN gateway function	It routes CAN communication by referring to a routing table.	CAN Gateway function		Y	Y
			Routing table	Routing table	Y	Y	
		OBD connection function	It sends data in CAN via OBD connector.	OBD connection function		Y	Y

7.2 Identification of threats with higher risks

Regarding the automotive security, the threat example of the automotive embedded system is poor at present. As a result, it is difficult to carry out risk evaluation for automotive embedded system design. Regarding embedded system, this clause describes extracted threats and performs risk analysis according to the framework of [ISO/IEC 15408-1]. Here, it is aimed that the risk analysis does not depend on the know-how of the security design. Therefore, in this Recommendation, the risk analysis method CVSS based risk scoring system (CRSS) [b-JASO TP15002] calculates a risk score for each threat for TOE.

Based on the TOE model and an overview of the module functions defined in clause 7.1, all the threats are identified and then higher-risk threats are extracted by means of the CRSS method.

Threats for TOE with a risk score more than a certain high value are shown in Table 2, based on the risk analysis method in Appendix I.

Table 2 – Threats for TOE with risk score more than a certain high value

#	Label	Who	When (phase)	Why	Where/What	Risk score
1	T.DoS-Functions-From-OBD-Device	Third party Maintenance factory staff	Normal operation Maintenance	Intentionally	For asset functions of VMG, it impersonates an OBD connector connection device, sends a huge amount of data, and interferes with this function.	6.6
2	T.Malfunction-Functions-From-OBD-Device	Third party Maintenance factory staff	Normal operation use/maintenance Maintenance	Intentionally	For asset functions of VMG, impersonates an OBD connector connection device, sends unauthorized data, and causes a malfunction of this functionality.	6.6
3	T.MissDoS-Functions-From-OBD-Device	Vehicle dealer staff Maintenance factory staff	Maintenance	Accidentally	For asset functions of VMG, it sends a huge amount of data or unauthorized data from OBD connector connection device by mistake, and causes a malfunction of this functionality.	6.6
4	T.DoS-Functions-From-ECU	Third party Maintenance factory staff	Normal operation/ use/maintenance Maintenance	Intentionally	For asset functions of VMG, it uses reverse engineering of the same product as the ECU firmware connected to CAN0-2, updates ECU firmware connected to CAN0-2 to an unauthorized firmware; in this way, it sends a huge amount of data from ECU connected to CAN1-5, and interferes with this functionality.	5.6

#	Label	Who	When (phase)	Why	Where/What	Risk score
5	T.Malfunction-Functions-From-ECU	Third party Maintenance factory staff	Normal operation/ use/maintenance Maintenance	Intentionally	For asset functions of VMG, it uses reverse engineering of the same product as the ECU firmware connected to CAN1-5, updates ECU firmware connected to CAN1-5 to an unauthorized firmware; in this way, it sends unauthorized data from ECU connected to CAN1-5, causes a malfunction of this functionality.	5.6
6	T.DoS-Functions-From-Mobile-Device	Third party	Normal operation/ use/maintenance	Intentionally	For asset functions of VMG, it impersonates a server, sends a huge amount of data to VMG from a mobile connection device, and interferes with this functionality.	9.4
7	T.Spoofing-Server_ToGet-Data	Third party	Normal operation/ use/ maintenance	Intentionally	For asset information of VMG, it sends a command to get VMG's asset information from mobile connection device by intercepting communication channel or by impersonating a mobile connection device. In this way, it receives VMG's asset information.	9.4
8	T.MissDoS-Functions-From-mobile-Device	Server administrator	Normal operation/ use/maintenance	Accidentally	For asset functions of VMG, the server sends, by misoperation, a huge amount of data or unauthorized data from mobile connection device, interferes with this functionality, and causes a malfunction of this functionality.	9.4

#	Label	Who	When (phase)	Why	Where/What	Risk score
9	T.Leaking-Mobile-Information-From-Mobile-Device	Owner/user Server administrator/ vehicle dealer staff Server administrator	Normal operation/ use vehicle delivery Normal operation/ use/maintenance	Accidentally	For asset information of VMG, from mobile connection device, it sends by misoperation, it sends a command to VMG to get VMG's protection asset (information), and get and leak VMG's protection asset (information).	8.5
10	T.MissUpdate-Mobile-Information-From-Mobile-Device	Owner/user Server administrator/ vehicle dealer staff Server administrator	Normal operation/use Vehicle delivery Normal operation/ use/ maintenance	Accidentally	For asset information of VMG, from mobile connection device, by misoperation, by mistake, it sends a command to VMG to update VMG's protection asset (information), and updates VMG's protection asset (information).	8.5
11	T.Malfunction-Functions-From-mobile-Device	Third party	Normal operation/ use/maintenance	Intentionally	For asset functions of VMG, from mobile connection device, it impersonates a server, sends unauthorized data, and causes a malfunction of this functionality.	8.5
12	T.Spoofing-Server_ToRewrite-Data	Third party	Normal operation/use	Intentionally	For protection asset (information) of VMG, from mobile connection device, impersonates a mobile connection device, sends a command to rewrite VMG's protection asset (information), and VMG's protection asset (information).	7.9

#	Label	Who	When (phase)	Why	Where/What	Risk score
13	T.DoS-Functions-From-Wi-Fi-Device	Third party	Normal operation/use/maintenance	Intentionally	For Wi-Fi connection function, it impersonates a Wi-Fi connection device, sends a huge amount of data, and interferes with this functionality.	7.8
14	T.Malfunction-Functions-From-Wi-Fi-Device	Third party	Normal operation/use/maintenance	Intentionally	For Wi-Fi connection function, it impersonates a Wi-Fi connection device, sends unauthorized data, and causes a malfunction of this functionality.	7.1
15	T.MissDoS-Functions-From-Wi-Fi-Device	Owner/user	Normal operation/use	Accidentally	For Wi-Fi connection function, by misoperation of Wi-Fi connection device or by malware infection of Wi-Fi connection device, it sends a huge amount of data or unauthorized data, interferes with this functionality, and causes a malfunction of this functionality	7.8
16	T.Spoofing-Wi-Fi-Device_ToGet-Wi-Fi-Information	Third party	Normal operation/use / maintenance	Intentionally	For Wi-Fi connection function, impersonates Wi-Fi connection device, and it sends a command to get Wi-Fi connection authentication information, and it exploits Wi-Fi connection authentication information.	7.8

#	Label	Who	When (phase)	Why	Where/What	Risk score
17	T.Spoofing-Wi-Fi-Device_ToRewrite-Wi-Fi-Information	Third party	Normal operation/use / maintenance	Intentionally	For Wi-Fi connection function, it impersonates a Wi-Fi connection device, sends a command to rewrite Wi-Fi connection authentication information, and rewrites Wi-Fi connection authentication information.	6.7
18	T.Leaking-Wi-Fi-Information-From-Wi-Fi-Device	Vehicle dealer staff Owner/user	Vehicle delivery Normal operation/use	Accidentally	For Wi-Fi connection authentication information, It sends a command to get a Wi-Fi connection authentication information, and gets and leaks Wi-Fi connection authentication information.	7.7
19	T.MissUpdate-Wi-Fi-Information-From-Wi-Fi-Device	Vehicle dealer staff Owner/user	Vehicle delivery Normal operation/ use	Accidentally	For Wi-Fi connection authentication information, it sends a command to rewrite a Wi-Fi connection authentication information, and rewrites Wi-Fi connection authentication information.	7.7

7.2.1 TOE security requirements

7.2.1.1 SR.integrity/availability protection of VMG functions through CAN communication

Integrity and availability of VMG functions shall be ensured against denial of service (DoS) and malfunction attacks from ECUs through CAN0-CAN2 communication (see Threat 4, 5).

Description

In CAN communication, the only CAN data to which the specified CAN identifiers (IDs) are given shall be routed. If VMG receives a huge amount of communication packets and/or confirms access of irregular patterns from CAN0-CAN2 connection devices, it shall not perform abnormal operations.

7.2.1.2 SR.confidentiality protection of VMG data

The communication contents between VMG and the server shall be protected in terms of confidentiality in the way that a third party cannot read them (see Threat 7, 16, 17).

7.2.1.3 SR.integrity/availability protection of VMG functions through mobile communication

Integrity and availability of VMG functions shall be ensured against DoS and malfunction attacks from mobile devices through mobile communication (see Threat 6, 7, 8, 9, 10, 11, 12).

Description

In communication with a mobile connection device, VMG shall confirm if the communicating party is an authorized mobile connection device. VMG shall be protected from impersonation of the server when receiving unauthorized/abnormal data through mobile communication. If VMG receives a huge amount of communication packets from mobile connection devices and/or confirms access of irregular patterns from mobile connection device, it shall not perform abnormal operations. Furthermore, VMG shall confirm the consistency and the frequency of transmission between commands sent from mobile connection devices.

7.2.1.4 SR.FaultTolerance of VMG functions

The functions of VMG shall continue their intended operations, possibly at a reduced level in the presence of something irregular due to attacks (see Threat 1, 2, 3, 4, 5, 6, 8, 11, 15).

7.2.1.5 SR.integrity/availability protection of VMG functions through OBD

Integrity and availability of VMG functions shall be ensured against DoS and malfunction attacks from OBD connection devices through an OBD connector (see Threat 1, 2, 3).

Description

Regarding CAN connection via the OBD connector, only specified devices are allowed to access ECUs. VMG shall be protected from impersonation of OBD connection devices when receiving unauthorized/abnormal data through the OBD connector. If VMG receives a huge amount of communication or unauthorized commands from OBD connection devices, it shall not perform abnormal operations.

7.2.1.6 SR.confidentiality/integrity/availability protection of VMG through Wi-Fi communication

VMG shall be protected from impersonation of Wi-Fi communication devices, when receiving unauthorized/abnormal data through Wi-Fi communication (see Threat 13, 14, 15, 16, 17, 18, 19).

Description

In communication with a Wi-Fi device, VMG shall confirm if the device has been registered in advance. If VMG receives a huge amount of communication packets from Wi-Fi devices and/or confirms access of irregular patterns from a Wi-Fi device, it shall not perform abnormal operations.

7.2.2 Security requirements for the operational environment of TOE from IT perspective

7.2.2.1 SRE.ECU protection

ECU module shall be protected against analysis of ECU firmware by means of obfuscation of the module. ECU shall be protected against attacks using unauthorized sensor data. ECU shall be physically protected against attacks by means of unauthorized replacement of ECU (see Threat 4, 5).

7.2.2.2 SRE.CAN communication protection

CAN communications shall be protected against analysis of CAN communication protocol by means of scrambling operations (lightweight operations such as bit flipping, etc.) on the CAN payload data. CAN shall be physically protected against attacks by means of clipping to CAN wiring by a malicious third party (see Threat 4, 5).

7.2.2.3 SRE.Mobile communication network protection

The mobile communication network that VMG uses to communicate with the server shall be protected against attacks from unauthorized devises. The network configuration information shall be protected in terms of confidentiality. The network shall be monitored to detect attacks (see Threat 6, 7, 11, 12).

7.2.2.4 SRE.Wireless communication protection

Wireless communications shall be protected against analysis of the wireless communication protocol by storing the only minimum data in the payload of the communication packet, or by scrambling payload data by means of lightweight operations such as bit flipping, etc. (see Threat 7, 12, 16, 17).

7.2.3 Security requirements for the operation environment from Non-IT operation/management perspective

7.2.3.1 SREN.Caution

It shall be noted that an attack on the in-vehicle system is a criminal act. Furthermore, the sale of the product to help the crime shall be restrained (see Threat 1, 2, 4, 5, 6, 7, 11, 12, 13, 14, 16, 17).

7.2.3.2 SREN.NetworkServicer

The server administrator shall prevent stored data from being leaked or tampered by means of an inappropriate management of the server (see Threat 7, 8).

7.2.3.3 SREN.OBD tool protection

OBD tools connect to a vehicle shall be protected against unauthorized usage by means of secure management. In addition, the operation methods of tools connected to a vehicle shall be confirmed before the operation (see Threat 3).

7.2.3.4 SREN.User

When users use a vehicle, they shall be informed about the required precautions.

Description

A vehicle shall be locked to prevent invasions of a third party if a user is away from it. A vehicle shall be parked in a place where a third party cannot easily approach it, if it is not in use. The user shall confirm that an unidentified device is not present before using a vehicle. The user shall be careful when he or she connects commercial products to the OBD connector that is an interface for the maintenance (see Threat 1, 2, 4, 5, 13, 14, 16, 17).

7.2.3.5 SREN.VirusScan

Devices connected to the system via mobile/Wi-Fi connections are regularly scanned (see Threat 9, 10, 15, 18, 19).

7.2.3.6 SREN.Wireless-Device protection

The concerned person shall confirm how to operate the device connected via mobile/Wi-Fi before

the operations. In addition, the concerned person shall be careful to prevent the leakage of the password of the devices connected via Wi-Fi and commands (see Threat 9, 10, 12, 13, 14, 16, 17, 18, 19).

7.2.3.7 SREN.Wireless-Display

Users using a Wi-Fi/mobile connection device shall confirm whether or not to send "get/write" commands of VMG asset information by making a choice on the display of the device (see Threat 9, 10, 18, 19).

7.3 Security controls

Based on the security requirements, this clause provides security controls that satisfy the security requirements especially from IT perspectives.

7.3.1 SC.Trusted boot

As a countermeasure against analysis (e.g. tampering) of the original program module in ECU, it is recommended for ECU to implement self-check mechanisms of its software by using the boot security protection mechanism of the hardware security module (HSM) at every boot sequence of ECU.

Corresponding security requirement

- SRE.ECU protection

7.3.2 SC.Message verification

Against the attacks of tampering, eavesdropping and replaying, the message verification method is effective to retain authenticity of entities and integrity of messages.

There are two methods which are appropriate for this purpose: the first is with digital signature (digital signature method), and the second is with message authentication code (MAC).

Meanwhile, for practical implementations of ECUs in a vehicle, cryptographic capabilities of devices differ depending on the vehicles. For example, a luxury vehicle might have HSMs for all ECUs in the vehicle, while a popular vehicle might have HSMs for only a part of ECUs. Besides, there are differences of cryptographic capabilities depending on the types of the applied HSMs.

Therefore, the security architecture needs to take into account the differences of security capabilities among vehicles. Namely, this Recommendation applies the digital signature method based on [ITU-T X.509] for the message verification of vehicles with asymmetric crypto algorithm (e.g. a trusted platform module (TPM). On the other hand, for vehicles without asymmetric crypto algorithm (e.g. HSM and smart-card), this Recommendation applies MAC for message verification. For the details of the communication protocol including the message verification, see clause 8.

Corresponding security requirements

- SR.confidentiality protection of VMG data;
- SR.confidentiality/integrity/availability protection of VMG through Wi-Fi communication;
- SRE.Mobile communication network protection;
- SRE.Wireless communication protection.

7.3.3 SC.Authentication of communication entity

In order to avoid impersonation of communication entities (i.e. impersonation of ECU, VMG and the update server), those entities are recommended to authenticate each other at the beginning of every communication. This security control should be implemented under the transport layer, and secure software update procedures defined in this Recommendation should be secured by the lower

layer function. As a specific countermeasure for authentication of communication entities, both client and server authentication using the secure socket layer/transport layer security (SSL/TLS) is effective under the third-party certification authority (CA).

Corresponding security requirements

- SR.confidentiality/integrity/availability protection of VMG through Wi-Fi communication;
- SRE.Mobile communication network protection;
- SRE.Wireless communication protection.

7.3.4 SC.Message filtering

As an example of DoS attacks against VMG, an ECU is compromised by an attacker and massively sends forged messages to VMG to improperly consume computational power of them. In order to reduce the security impact by such DoS attacks, message filtering technique is one of the effective methods. It is recommended for VMG to filter out irrelevant messages according to sender ID, message type, size, frequency, etc., or a combination of all of them.

Corresponding security requirements

- SR.integrity/availability protection of VMG functions through CAN communication;
- SR.integrity/availability protection of VMG functions through mobile communication;
- SR.integrity/availability protection of VMG functions through OBD.

7.3.5 SC.FaultTolerance of VMG functions

Suppliers of VMG are strongly recommended to implement VMG software with fail-safe design so that VMG can continue its intended operations in the presence of something irregular due to attacks. Particularly, VMG monitors the operation status and if something irregular is detected, a measure (reboot, etc.) is taken so that it recovers to a normal state. If it is not possible to recover, it informs the driver of the issue and safely suspends the operation.

Corresponding security requirement

- SR.FaultTolerance of VMG functions.

8 Specification of secure software update procedure

This clause defines a practical procedure for software update with security functions. Note that this Recommendation does not specify transport protocols (e.g. transport layer protocol, application layer protocols (hypertext transfer protocol (HTTP), and hypertext transfer protocol secure (HTTPS), etc.) and tunnelling protocol, etc.), but it does specify application data format of the messages.

As described in clause 7.3.2, the procedure needs to take into account the diversity of security capabilities among vehicles (ECUs). Therefore in this Recommendation, ECUs with asymmetric crypto algorithm apply the digital signature method (in clause 8.1.1), while ECUs without asymmetric crypto algorithm apply the MAC method (in clause 8.1.2) for secure message exchange. Car manufactures or suppliers can select an adequate method out of the two security methods according to the security capabilities of their vehicles.

8.1 General message format with security functions

This clause introduces a general message format with security functions including authentication method of message sender and verification of integrity of a message. In terms of integrity and authenticity technique, the digital signature method with public key algorithm and message authentication code (MAC) with shared key algorithm can be applied. In the secure software update procedure, every message is constructed in one of the security methods as follows.

8.1.1 Digital signature method

For ECUs with asymmetric crypto capability by HSM (e.g. TPM), the digital signature method based on [ITU-T X.509] can be applied for authentication of entities and verification of a message integrity.

8.1.2 MAC method

Since the shared key algorithm requires less processing load than the public key algorithm, the shared key algorithm is suitable for devices with a low processing capacity. However, in the shared key algorithm, the sender and receiver use the same key, thus a large number of devices use the same key. This operation requires the update of keys in all devices in the system once the shared key is leaked. Besides, since the shared key itself does not provide authenticity of the sender, each message needs to contain a device ID of the sender, which presupposes that the ID in the device is not improperly manipulated.

8.2 Protocol definition and data format

The application data format is dedicated to deliver messages only regarding software update, which is enclosed in the general message format described in the previous clause. This clause firstly defines message types applied in the software update procedure, and secondly it introduces specifications of the message types. The messages are described with an extended markup language (XML) format.

8.2.1 Protocol overview

Based on the model of software update procedure discussed in clause 6, the messages are categorized into several types according to their objectives, as depicted in Figure 4. The procedure is initiated by VMG to diagnose ECUs and collect the status of software in ECUs by a diagnose message.

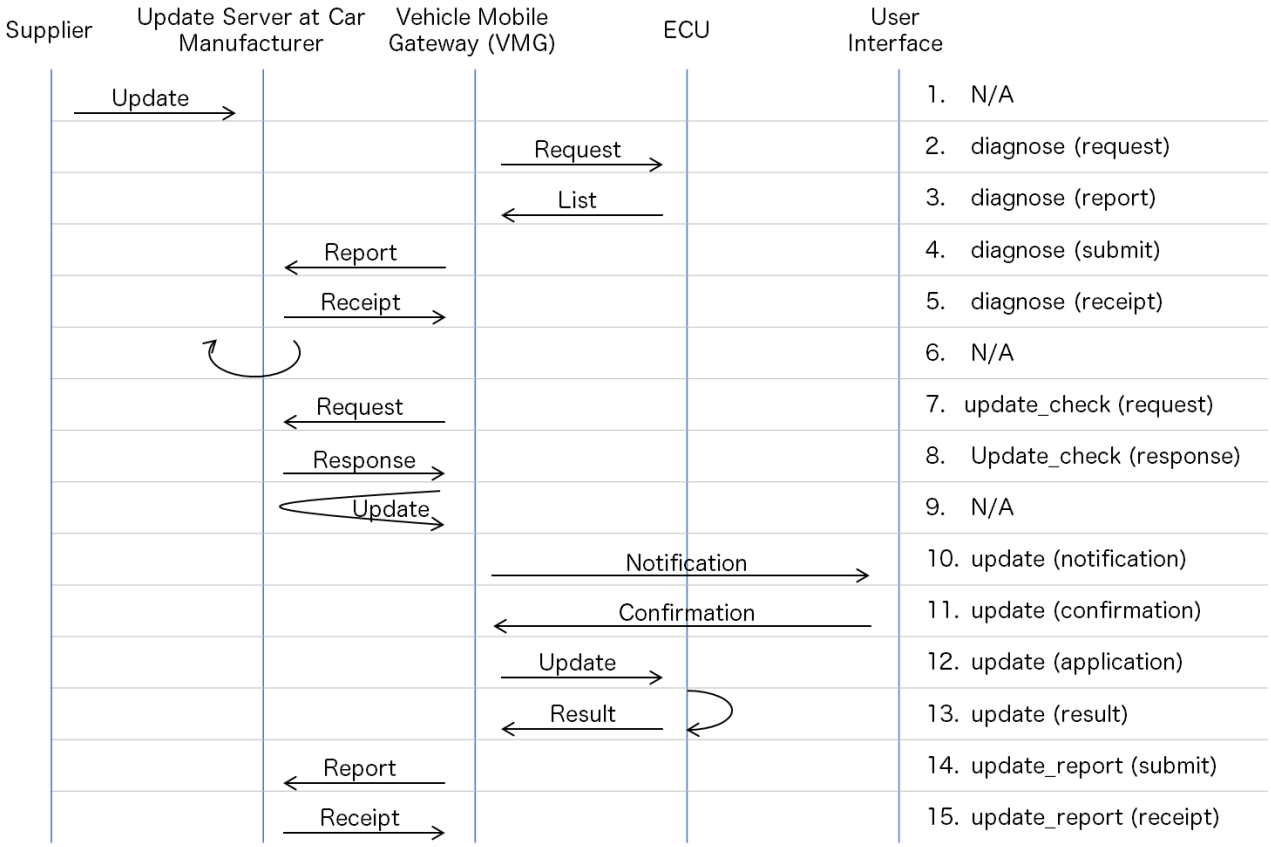

Figure 4 – Definition of message types

In steps 2, 3, 4 and 5, since the purpose of the messages are request and report of diagnoses of software status of each ECU, the messages are categorized as "diagnose". In the same manner, the messages in steps 7 and 8 are categorized as "update_check". Steps 10, 11, 12 and 13 are "update", which are messages to confirm and apply the updates. Finally, the results of the updates are submitted by "update_report" messages in steps 14 and 15. The types, subtypes and codes of the messages are shown in Table 3.

Table 3 – Message types

Type	Subtype	From	To	Purpose
diagnose	request	VMG	ECU	Request of diagnose of software status
	report	ECU	VMG	Result of diagnose including software status
	submit	VMG	Usvr	Report of results of ECUs in a vehicle
	receipt	Usvr	VMG	Receipt for submit of diagnose report
update_check	request	VMG	Usvr	Request of update module
	response	Usvr	VMG	Update module provided
update	notification	VMG	UI	Notification message to introduce update for the driver
	confirmation	UI	VMG	Confirmation message from the driver to apply update
	application	VMG	ECU	Request message including update module
	result	ECU	VMG	Result of application of the update module
update_report	submit	VMG	Usvr	Report of application of the update
	receipt	Usvr	VMG	Receipt of the report

* Usvr: Update server

* UI: User interface

8.2.2 diagnose messages

With the purpose of determining the necessary update modules for a vehicle, a diagnose message is used between an update server and VMG to upload software information from vehicles to the update server. A diagnose message is also exchanged between VMG and ECUs to gather software information of ECUs in the vehicle. There are three subtypes of diagnose message: request, report and submit, that are applied to these usages.

8.2.2.1 diagnose (request) message

The diagnose (request) is the initial message of the whole software update procedure, which is sent from VMG to ECUs in a vehicle, to request ECUs to submit software status information of ECUs.

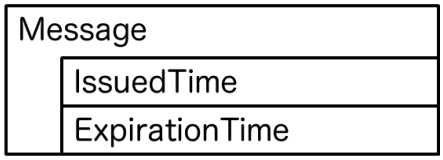

Figure 5 – Structure of diagnose (request) message

Table 4 – Elements of diagnose (request) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".
	version	The version number of the message sender.
	type	Message type (always "diagnose").
	subtype	Message subtype (always "request").
	sessionid	Session ID is a random global user ID (GUID) associated with the diagnose session. An identical session ID is applied to a set of diagnose request, report, submit and receipt messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of the device that generated this message.
	messageid	Message ID is a random GUID associated with an individual message.
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 5 – Example of diagnose (request) message

```
<Message protocol="1.0" version="1.0.2" type="diagnose"
subtype="request" sessionid="{7316A97D-8C04-428B-B498-
0F51087A1093}" messageid="{2E255A59-B875-4347-90CA-92326BF45BEF}"
trustlevel="3">
  <IssuedTime>1903-07-01T00:00:00Z</IssuedTime>
  <ExpirationTime>1903-07-01T00:00:00Z</ExpirationTime>
</message>
```

8.2.2.2 diagnose (report) message

In response to a request of diagnose from VMG, ECU sends diagnose (report) message which includes software status information of ECU. The software information is a set of version and hash value of the software, which is stored in a particular structure.

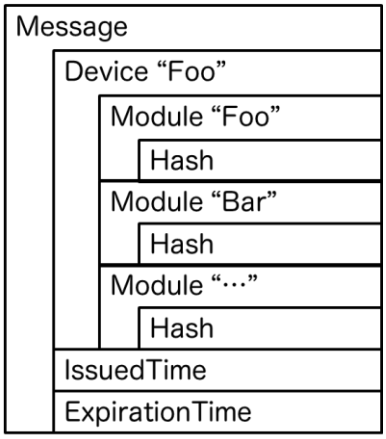

Figure 6 – Structure of diagnose (report) message

Table 6 – Elements of diagnose (report) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".
	version	The version number of the message sender.
	type	Message type (always 'diagnose").
	subtype	Message subtype (always "report").
	sessionid	Session ID is a random GUID associated with the diagnose session. An identical session ID is applied to a set of diagnose request, report, submit and receipt messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of the device that generated this message.
	ownerid	Owner ID provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.
Device	-	Container of device information. It contains multiple module elements.
	name	Name of the device, if any.
	type	Type name of the device, such as "Power management ECU", "Seat belt control ECU", etc.
	model	Model name of the device.
	deviceid	Device id defined by a car manufacturer/supplier.
Module	-	Container of module information, which contains a hash element.
	moduleid	Module id is a unique id provided by a car manufacturer/a supplier.
	version	Version of this software module.
	nextversion	The version of the module update in progress, which is mainly used for sending a report message during an update.

Element	Attribute in element	Description
Hash	-	Hash is a container of a hash value and information of its hash algorithm.
	algorithm	Algorithm of the hash function (e.g. SHA-3, SHA-256, etc.).
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 7 – Example of diagnose (report) message

```
<message protocol="1.0" version="1.0.2" type="diagnose"
subtype="report" sessionid="{7316A97D-8C04-428B-B498-
0F51087A1093}" ownerid="ownerid987239487" messageid="{66E6F81E-
F293-4531-B2FC-A93F177373AA}" trustlevel="3">
    <Device name="device1" type="ECU" model="model1"
deviceid="did0987234">
        <Module moduleid="{66E6F81E-F293-4531-B2FC-A93F177373AA}"
version="1.3.23.0" nextversion=""
            <Hash algorithm="SHA-256">hash data here</Hash>
        </Module>
    </Device>
  <IssuedTime>1903-07-01T00:00:00Z</IssuedTime>
  <ExpirationTime>1903-07-01T00:00:00Z</ExpirationTime>
</message>
```

8.2.2.3 diagnose (submit) message

After collecting the diagnose results (response) from ECUs, VMG submits a list of software information to the update server in the manufacture (or garage). The diagnose (submit) message includes the identity of the vehicle (vid) and a list of software information that is extracted from diagnose (report) messages.

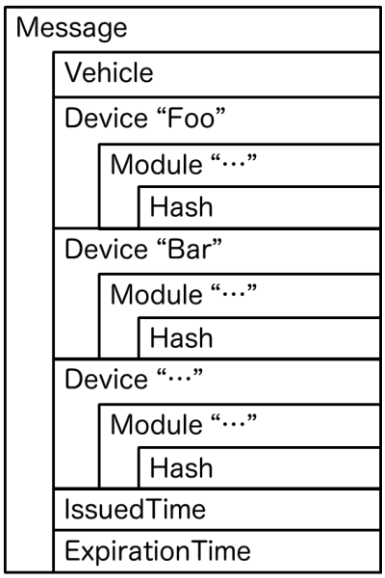

Figure 7 – Structure of diagnose (submit) message

Table 8 – Elements of diagnose (submit) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always '1.0".
	version	The version number of the message sender.
	type	Message type (always "diagnose").
	subtype	Message subtype (always "submit").
	sessionid	Session ID is a random GUID associated with the diagnose session. An identical session ID is applied to a set of diagnose request, report, submit and receipt messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of the device that generated this message.
	ownerid	Owner ID provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.
Vehicle	-	Container of vehicle information. It contains multiple module elements.
	name	Name of the vehicle, if any.
	model	Type name of the device provided by the car manufacturer.
	modelid	Model name of the vehicle.
	vehicleid	Vehicle id defined by a car manufacturer/supplier.
Device	-	Container of device information. It contains multiple module elements.
	name	Name of the device, if any.
	type	Type name of the device, such as "Power management ECU", "Seat belt control ECU", etc.
	model	Model name of the device.
	deviceid	Device id defined by a car manufacturer/supplier.
Module	-	Container of module information, which contains a hash element.
	moduleid	Module id is a unique id provided by a car manufacturer/a supplier.
	version	Version of this software module.
	nextversion	The version of the module update in progress, which is mainly used for sending a response message during an update.
Hash	-	Hash is a container of a hash value and information of its hash algorithm.
	algorithm	Algorithm of the hash function (e.g. SHA-3, SHA-256, etc.).
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 9 – Example of diagnose (submit) message

```
<message protocol="1.0" version="1.0.2" type="diagnose"
subtype="submit" sessionid="{7316A97D-8C04-428B-B498-
0F51087A1093}" ownerid="oid987239487" messageid="{BBCE3B0B-2A10-
443A-97D0-EF4650457422}" trustlevel="3">
```

```
    <Vehicle name="vehicleName" model="modelName"
modelid="mid34987130" vehicleid="vid0987234"/>
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{66E6F81E-F293-4531-B2FC-
A93F177373AA }" version="1.3.23.0" nextversion=""/>
            <Hash algorithm="SHA-256">hash data here</Hash>
        </Module>
        <Module moduleid="{4D168B58-26FA-4157-9703-A431D99C8438}"
version="2.4.34.0" nextversion=""/>
            <Hash algorithm="SHA-256">hash data here</Hash>
        </Module>
    </Device>
    <Device name="device2" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{70628FDC-2282-4B2F-8A36-13445DED587A}"
version="3.5.45.0" nextversion=""/>
            <Hash algorithm="SHA-256">hash data here</Hash>
        </Module>
    </Device>
    <IssuedTime "1903-07-01T00:00:00Z"/>
    <ExpirationTime "1903-07-01T00:00:00Z"/>
</message>
```

8.2.2.4 diagnose (receipt) message

After the upload of a software information of a vehicle with diagnoses (submit) message, the update server sends back a receipt with diagnose (receipt) message so that the vehicle recognizes that the submission was successfully finished and that the vehicle can proceed to the next state (update_check).

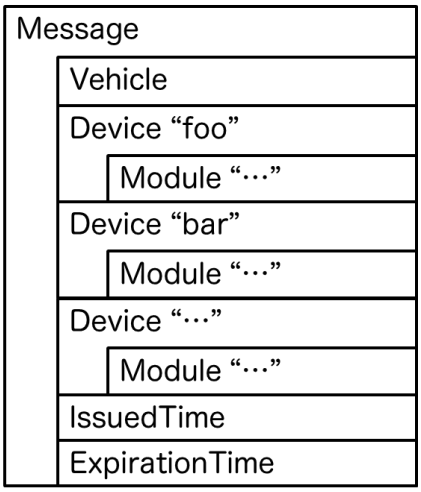

Figure 8 – Structure of diagnose (receipt) message

Table 10 – Elements of diagnose (receipt) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".

Element	Attribute in element	Description
	version	The version number of the message sender.
	type	Message type (always "diagnose").
	subtype	Message subtype (always 'receipt").
	sessionid	Session ID is a random GUID associated with the diagnose session. An identical session ID is applied to a set of diagnose request, report, submit and receipt messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of the device that generated this message.
	ownerid	Owner ID provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.
Vehicle	-	Container of vehicle information. It contains multiple module elements.
	name	Name of the vehicle, if any.
	model	Type name of the device provided by the car manufacturer.
	modelid	Model name of the vehicle.
	vehicleid	Vehicle id defined by a car manufacturer/supplier.
Device	-	Container of device information. It contains multiple module elements.
	name	Name of the device, if any.
	type	Type name of the device, such as "Power management ECU", "Seat belt control ECU", etc.
	model	Model name of the device.
	deviceid	Device id defined by a car manufacturer/supplier.
Module	-	Container of module information, which contains a hash element.
	moduleid	Module id is a unique id provided by a car manufacturer/a supplier.
	version	Version of this software module.
	nextversion	The version of the module update in progress, which is mainly used for sending a response message during an update.
	status	Acknowledgement of the report for this module.
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 11 – Example of diagnose (receipt) message

```
<message protocol="1.0" version="1.0.2" type="diagnose"
subtype="receipt" sessionid="{7316A97D-8C04-428B-B498-
0F51087A1093}" ownerid="oid987239487" messageid="{E313159C-2081-
4A10-B61D-4F81D074D54F}" trustlevel="3">
    <Vehicle name="vehicleName" model="modelName"
modelid="mid34987130" vehicleid="vid0987234"/>
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{1F6EDD6C-17D4-461B-8403-1E240E26464E}"
version="" nextversion="1.3.23.0" status="ok"/>
```

```
        <Module moduleid="{4D168B58-26FA-4157-9703-A431D99C8438}"
version="2.4.34.0" nextversion="" status="ok"/>
    </Device>
    <Device name="device2" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{70628FDC-2282-4B2F-8A36-13445DED587A}"
version="3.5.45.0" nextversion="" status="ok"/>
    </Device>
    <IssuedTime "1903-07-01T00:00:00Z"/>
    <ExpirationTime "1903-07-01T00:00:00Z"/>
</message>
```

8.2.3 update_check messages

After a software information is uploaded to the update server by a diagnose message, the update server starts the analysis to determine the necessary update modules for the vehicle, which might take a long time. The update_check message is periodically applied to inquire about the decision of the update server. There are two subtypes of update_check message, request and response, which are transferred between VMG and an update server.

8.2.3.1 update_check (request) message

The update_check (request) message is transferred from VMG to an update server to check for the necessity of the updates. This message includes information of modules to be inspected, which are quite similar to diagnose (receipt) message.

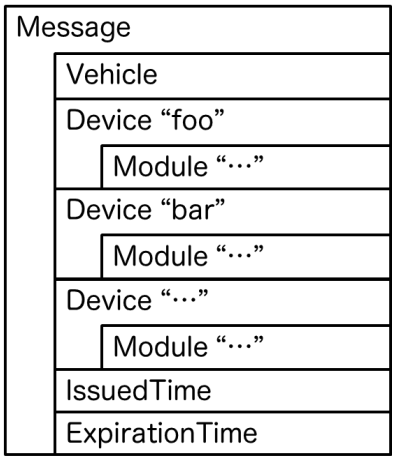

Figure 9 – Structure of update_check (request) message

Table 12 –Elements of update_check (request) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".
	version	The version number of the message sender.
	type	Message type (always "update_check").
	subtype	Message subtype (always "request").

Element	Attribute in element	Description
	sessionid	Session ID is a random GUID associated with the update_check session. An identical session ID is applied to a set of update_check request and response messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of the device that generated this message.
	ownerid	Owner ID is provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.
Vehicle	-	Container of vehicle information. It contains multiple module elements.
	name	Name of the vehicle, if any.
	model	Type name of the device provided by the car manufacturer.
	modelid	Model name of the vehicle.
	vehicleid	Vehicle id defined by a car manufacturer/supplier.
Device	-	Container of device information. It contains multiple module elements.
	name	Name of the device, if any.
	type	Type name of the device, such as "Power management ECU", "Seat belt control ECU", etc.
	model	Model name of the device.
	deviceid	Device id defined by a car manufacturer/supplier.
Module	-	Container of module information, which contains a hash element.
	moduleid	Module id is a unique id provided by a car manufacturer/a supplier.
	version	Version of this software module.
	nextversion	The version of the module update in progress, which is mainly used for sending a response message during an update.
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 13 – Example of update_check (request) message

```
<message protocol="1.0" version="1.0.2" type="update_check"
subtype="request" sessionid="{19622672-A025-4500-B26A-
BB626BC61C62}" ownerid="oid987239487" messageid="{4604A6C9-F72F-
452B-ABA5-94168CCD8FD6}" trustlevel="3">
    <Vehicle name="vehicleName" model="modelName"
modelid="mid34987130" vehicleid="vid0987234"/>
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{1F6EDD6C-17D4-461B-8403-1E240E26464E}"
version="" nextversion="1.3.23.0"/>
        <Module moduleid="{4D168B58-26FA-4157-9703-A431D99C8438}"
version="2.4.34.0" nextversion=""/>
    </Device>
    <Device name="device2" type="ECU" model="model1"
id="did0987234">
```

```
        <Module moduleid="{70628FDC-2282-4B2F-8A36-13445DED587A}"
version="3.5.45.0" nextversion=""/>
    </Device>
    <IssuedTime "1903-07-01T00:00:00Z"/>
    <ExpirationTime "1903-07-01T00:00:00Z"/>
</message>
```

8.2.3.2 update_check (response) message

In response to update_check (request) message, the update server sends back the result of the inspection. If there are necessary updates for any of modules in the vehicle, the update_check (response) message downloads URLs to obtain the update modules. Note that an update message does not contain a binary file of the update module itself, while VMG downloads it with another connection based on the resource information in the update_check (response) message.

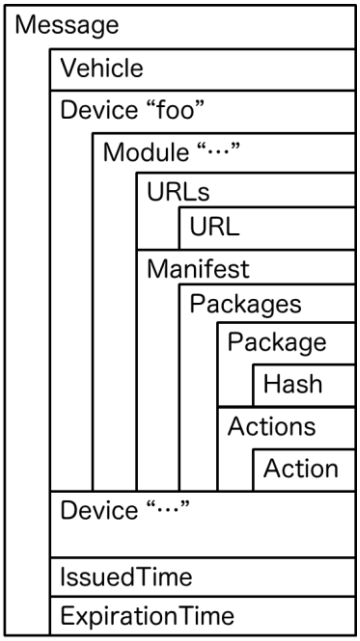

Figure 10 – Structure of update_check (response) message

Table 14 – Elements of update_check (response) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".
	version	The version number of the message sender.
	type	Message type (always "update_check").
	subtype	Message subtype (always "response").
	sessionid	Session ID is a random GUID associated with the update_check session. An identical session ID is applied to a set of update_check request and response messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of the device that generated this message.
	ownerid	Owner ID provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.

Element	Attribute in element	Description
Vehicle	-	Container of vehicle information. It contains multiple module elements.
	name	Name of the vehicle, if any.
	model	Type name of the device provided by the car manufacturer.
	modelid	Model name of the vehicle.
	vehicleid	Vehicle id defined by a car manufacturer/supplier.
Device	-	Container of device information. It contains multiple module elements.
	name	Name of the device, if any.
	type	Type name of the device, such as "Power management ECU", "Seat belt control ECU", etc.
	model	Model name of the device.
	deviceid	Device id defined by a car manufacturer/supplier.
Module	-	Container of module information, which contains a hash element.
	moduleid	Module id is a unique id provided by a car manufacturer/a supplier.
	version	Version of this software module.
	nextversion	The version of the module update in progress, which is mainly used for sending a response message during an update.
	status	Status of the inspection of update. "noupdate" is set if there are no updates, while "ok" is set if there are any updates for this module.
URLs	-	Container of URL elements if there are any updates. This element is contained in a module element where its status is ok.
URL	-	URL of update file.
	codebase	Location of the update file.
Manifest	-	Describes the module needed to be installed, and the actions needed to be taken with those files.
	version	Specific newer version number of this software module.
Packages	-	A set of files needed to be installed. Contains no attribution. Contains one or more package child elements.
Package	-	A single file to be installed for the module.
	name	Describes the filename of the update module.
	size	Contains the size in bytes of the update module.
Hash	-	Container of a hash value and information of its hash algorithm.
	algorithm	Algorithm of the hash function (e.g. SHA-3, SHA-256, etc.).
Actions	-	Actions to be performed to install the module once all required files in the packages element have been successfully downloaded.
Action	-	A single action to perform as part of the install process.
	event	A fixed string denoting when this action should be run. One of "preinstall", "install", "postinstall" and "update".
	arguments	Arguments to be passed to the installation process.
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 15 – Example of update_check (response) message

```
<message protocol="1.0" version="1.0.2" type="update_check"
subtype="response " sessionid="{19622672-A025-4500-B26A-
BB626BC61C62}" ownerid="oid987239487" messageid="{4604A6C9-F72F-
452B-ABA5-94168CCD8FD6}" trustlevel="3">
    <Vehicle name="vehicleName" model="modelName"
modelid="mid34987130" vehicleid="vid0987234"/>
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{1F6EDD6C-17D4-461B-8403-1E240E26464E}"
version="1.3.23.0" nextversion="" status="ok">
            <Urls>
                <Url
codebase="http://update.server/this/is/an/example/url/"/>
            </Urls>
            <Manifest version="1.4.0">
                <Packages>
                    <Package name="module1.bin" size="589">
                        <Hash algorithm="SHA-256">hash data
here</Hash>
                    </Package>
                </Packages>
                <Actions>
                    <Action arguments="--argument-for-
installation" event="install"/>
                </Actions>
            </Manifest>
        </Module>
        <Module moduleid="{4D168B58-26FA-4157-9703-A431D99C8438}"
version="2.4.34.0" nextversion="" status="noupdate">
        </Module>
    </Device>
    <Device name="device2" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{70628FDC-2282-4B2F-8A36-13445DED587A}"
version="3.5.45.0" nextversion="" status="noupdate">
        </Module>
    </Device>
    <IssuedTime "1903-07-01T00:00:00Z"/>
    <ExpirationTime "1903-07-01T00:00:00Z"/>
</message>
```

8.2.4 update messages

When an update is found for a device, VMG sends a notification to the user interface device by update (notification) message prior to practically applying the update to the device. If the driver confirms and accepts the update, an update (confirmation) message is sent from the user interface to VMG so that VMG proceeds with the applying process. For the application of the update module to the device, an update (application) message is used. Afterwards, the result of the application is reported by an update (result) message.

8.2.4.1 update (notification) message

An update (notification) message is used to notify the driver of the update. The message format is similar to an update_check (response) message so that the user interface can provide the driver of the updates with detailed information.

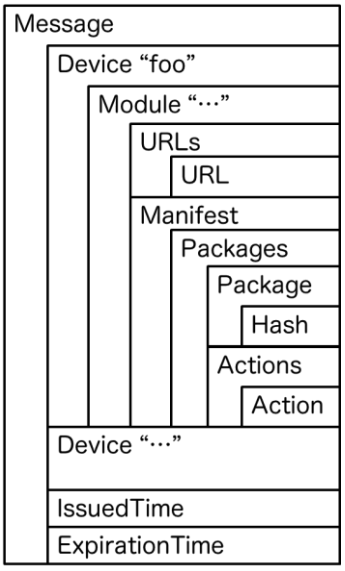

Figure 11 – Structure of update (notification) message

Table 16 – Elements of update (notification) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".
	version	The version number of the message sender.
	type	Message type (always "update").
	subtype	Message subtype (always "notification").
	sessionid	Session ID is a random GUID associated with the update session. An identical session ID is applied to a set of update notification, confirmation, application and result messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of the device that generated this message.
	ownerid	Owner ID provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.
Device	-	Container of device information. It contains multiple module elements.
	name	Name of the device, if any.
	type	Type name of the device, such as "Power management ECU", "Seat belt control ECU", etc.
	model	Model name of the device.
	deviceid	Device id defined by a car manufacturer/supplier.
Module	-	Container of module information, which contains a hash element.

Element	Attribute in element	Description
	moduleid	Module id is a unique id provided by a car manufacturer/a supplier.
	version	Version of this software module.
	nextversion	The version of the module update in progress, which is mainly used for sending a response message during an update.
	status	Status of the inspection of update. Always "ok" is set.
URLs	-	Container of URL elements if there are any updates. This element is contained in a module element where its status is ok.
URL	-	URL of update file.
	codebase	Location of the update file.
Manifest	-	Describes the module needed to be installed, and the actions needed to be taken with those files.
	version	Specific newer version number of this software module.
Packages	-	A set of files needed to be installed. Contains no attribution. Contains one or more package child elements.
Package	-	A single file to be installed for the module.
	name	Describes the filename of the update module.
	size	Contains the size in bytes of the update module.
Hash	-	Container of a hash value and information of its hash algorithm.
	algorithm	Algorithm of the hash function (e.g. SHA-3, SHA-256, etc.).
Actions	-	Actions to be performed to install the module once all required files in the packages element have been successfully downloaded.
Action	-	A single action to perform as part of the install process.
	event	A fixed string denoting when this action should be run. One of "preinstall", "install", "postinstall" and "update".
	arguments	Arguments to be passed to the installation process.
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 17 – Example of update (notification) message

```
<message protocol="1.0" version="1.0.2" type="update"
subtype="notification" sessionid="{BC5E12A1-3A6A-49F9-A3CA-
230740B745DE}" ownerid="oid987239487" messageid="{C7BF700E-D179-
4849-92F5-37ECD87740A0}" trustlevel="3">
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{1F6EDD6C-17D4-461B-8403-1E240E26464E}"
version="1.3.23.0" nextversion="" status="ok">
            <Urls>
                <Url codebase="http://itu-
t.int/this/is/an/example/url/"/>
            </Urls>
            <Manifest version="1.4.0">
                <Packages>
                    <Package name="module1.bin" size="589">
```

```
                        <Hash algorithm="SHA-256">hash data
here</Hash>
                    </Package>
                </Packages>
                <Actions>
                    <Action arguments="--argument-for-
installation" event="install"/>
                </Actions>
            </Manifest>
        </Module>
    </Device>
    <IssuedTime "1903-07-01T00:00:00Z"/>
    <ExpirationTime "1903-07-01T00:00:00Z"/>
</message>
```

8.2.4.2 update (confirmation) message

Once a driver prefers to apply the update, an update (confirmation) message is transferred from the user interface to VMG. Note that if there are multiple update modules, those updates have to be applied at one time (i.e. a driver cannot choose one or more updates to be applied) because each update module might have a sensitive mutual dependency.

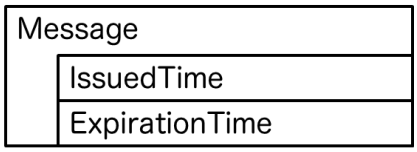

Figure 12 – Structure of update (confirmation) message

Table 18 – Elements of update (confirmation)

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".
	version	The version number of the message sender.
	type	Message type (always "update").
	subtype	Message subtype (always "confirmation").
	sessionid	Session ID is a random GUID associated with the update session. An identical session ID is applied to a set of update notification, confirmation, application and result messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of the device that generated this message.
	ownerid	Owner ID provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.
	status	A driver's preference for the application of the updates. 'yes" or "no" is set.
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 19 – Example of update (confirmation) message

```
<message protocol="1.0" version="1.0.2" type="update"
subtype="confirmation" sessionid="{BC5E12A1-3A6A-49F9-A3CA-
230740B745DE}" ownerid="oid987239487" messageid="{939C1A21-FA3C-
4080-9006-09094B85E698}" trustlevel="3">
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{1F6EDD6C-17D4-461B-8403-1E240E26464E}"
version="1.3.23.0" nextversion="1.4.0" status="ok">
        </Module>
    </Device>
    <IssuedTime "1903-07-01T00:00:00Z"/>
    <ExpirationTime "1903-07-01T00:00:00Z"/>
</message>
```

8.2.4.3 update (application) message

Based on the driver's confirmation, VMG sends update (application) messages to ECUs that need to be updated. In contrast to update_check (response) message which just includes URL of update modules, an update (application) message does include a binary file of the update to be applied to ECU, taking into account the insufficient capability of ECUs.

There are ECUs that do not mount enough memory resource for caching a whole update module at one time. For those ECUs, the stream updating technology might be required, which applies update modules by streaming fragmented data.

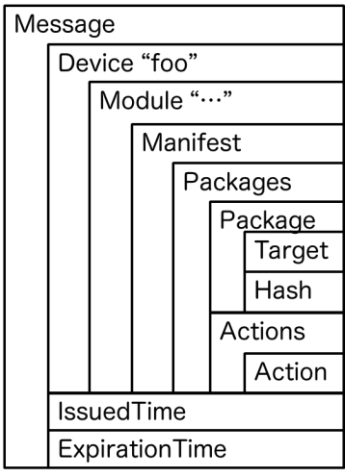

Figure 13 – Structure of update (application) message

Table 20 – Elements of update (application) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".
	version	The version number of the message sender.
	type	Message type (always "update").
	subtype	Message subtype (always "application").

Element	Attribute in element	Description
	sessionid	Session ID is a random GUID associated with the update session. An identical session ID is applied to a set of update notification, confirmation, application and result messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of the device that generated this message.
	ownerid	Owner ID provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.
Device	-	Container of device information. It contains multiple module elements.
	name	Name of the device, if any.
	type	Type name of the device, such as "Power management ECU", "Seat belt control ECU", etc.
	model	Model name of the device.
	deviceid	Device id defined by a car manufacturer/supplier.
Module	-	Container of module information, which contains a hash element.
	moduleid	Module id is a unique id provided by a car manufacturer/a supplier.
	version	Version of this software module.
	nextversion	The version of the module update in progress, which is mainly used for sending a response message during an update.
Manifest	-	Describes the module needed to be installed, and the actions needed to be taken with those files.
	version	Specific newer version number of this software module.
Packages	-	A set of files needed to be installed. Contains no attribution. Contains one or more package child elements.
Package	-	A single file to be installed for the module.
	name	Describes the filename of the update module.
	size	Contains the size in bytes of the update module.
Target	-	Encoded binary data as an update module.
	encode	Specification of encodings ("base64Binary" or "hexBinary").
Hash	-	Container of a hash value and information of its hash algorithm.
	algorithm	Algorithm of the hash function (e.g. SHA-3, SHA-256, etc.).
Actions	-	Actions to be performed to install the module once all required files in the packages element have been successfully downloaded.
Action	-	A single action to perform as part of the install process.
	event	A fixed string denoting when this action should be run. One of "preinstall", "install", "postinstall" and "update".
	arguments	Arguments to be passed to the installation process.
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 21 – Example of update (application) message

```
<message protocol="1.0" version="1.0.2" type="update"
subtype="application" sessionid="{BC5E12A1-3A6A-49F9-A3CA-
230740B745DE}" ownerid="oid987239487" messageid="{EBB65A6F-F963-
4EBD-A1DA-DEFA9997AE6A}" trustlevel="3">
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{1F6EDD6C-17D4-461B-8403-1E240E26464E}"
version="1.3.23.0" nextversion="" status="ok">
            <Manifest version="1.4.0">
                <Packages>
                    <Package name="module1.bin" size="589">
                        <Target
encode="base64Binary">Vm0wd2QyUXlVWGxWV0d4V1YwZDRWMVl3WkRSV01WbDN
Xa1JTVjAxV2JETlhhMUpUVjBaS2RHVkdXbFpOYWtFeFZtcEJlRll5U2tWVWJHaG9U
V3N3ZUZadGNFZFpWMDE1VTJ0V1ZXSkhhRzlVVjNOM1pVWmtWMXBFVWxSTmF6RTBWM
nRvUjFWdFNrZFhiR2hhWVRKb1JGWldXbUZqVmtaMFVteHdWMDF...</Target>
                        <Hash algorithm="SHA-256">hash data
here</Hash>
                    </Package>
                </Packages>
                <Actions>
                    <Action arguments="--argument-for-
installation" event="install"/>
                </Actions>
            </Manifest>
        </Module>
    </Device>
    <IssuedTime "1903-07-01T00:00:00Z"/>
    <ExpirationTime "1903-07-01T00:00:00Z"/>
</message>
```

8.2.4.4 update (result) message

An update (result) message reports the result of the update process from the devices to VMG in response to the update (application) message. It contains module elements with status attributes which indicate the results of the application.

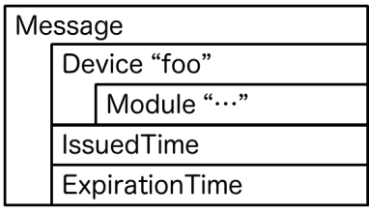

Figure 14 – Structure of update (result) message

Table 22 – Elements of update (result) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".

Element	Attribute in element	Description
	version	The version number of the message sender.
	type	Message type (always 'update").
	subtype	Message subtype (always "result")
	sessionid	Session ID is a random GUID associated with the update session. An identical session ID is applied to a set of update notification, confirmation, application and result messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of device that generated this message.
	ownerid	Owner ID provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.
Device	-	Container of device information. It contains multiple module elements.
	name	Name of the device, if any.
	type	Type name of the device, such as "Power management ECU", "Seat belt control ECU", etc.
	model	Model name of the device.
	deviceid	Device id defined by a car manufacturer/supplier.
Module	-	Container of module information, which contains a hash element.
	moduleid	Module id is a unique id provided by a car manufacturer/a supplier.
	version	Version of this software module.
	nextversion	The version of the module update in progress, which is mainly used for sending a response message during an update.
	status	Result of update process in ECU.
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 23 – Example of update (result) message

```
<message protocol="1.0" version="1.0.2" type="update"
subtype="result" sessionid="{BC5E12A1-3A6A-49F9-A3CA-
230740B745DE}" ownerid="oid987239487" messageid="{EBB65A6F-F963-
4EBD-A1DA-DEFA9997AE6A}" trustlevel="3">
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{1F6EDD6C-17D4-461B-8403-1E240E26464E}"
version="1.3.23.0" nextversion="" status="ok">
        </Module>
    </Device>
    <IssuedTime "1903-07-01T00:00:00Z"/>
    <ExpirationTime "1903-07-01T00:00:00Z"/>
</message>
```

8.2.5 update_report messages

As the final step of a sequence of update procedures, VMG submits all the collected reports of an update application in the devices to the update server so that the latter grasps and manages each vehicle from a remote site. VMG sends a report to the update server via update_report (submit) message. Finally, the update server sends a receipt of the report (update_report (receipt) message) to VMG so that it can recognize the end of the whole update process.

8.2.5.1 update_report (submit) message

After collecting application reports from the devices, VMG sends an update_report (submit) message to the update server. This message includes the results of the applications as well as the current status of software such as diagnose (submit) message.

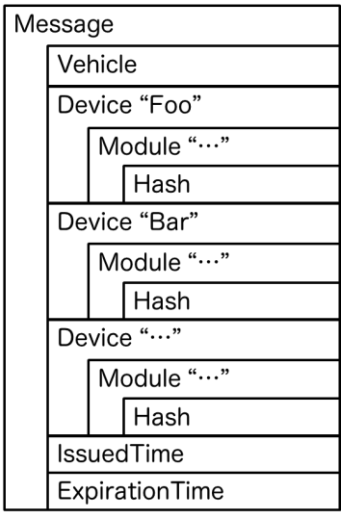

Figure 15 – Structure of update_report (submit) message

Table 24 – Elements of update_report (submit) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".
	version	The version number of the message sender.
	type	Message type (always "update_report").
	subtype	Message subtype (always "submit").
	sessionid	Session ID is a random GUID associated with the update_report session. An identical session ID is applied to a set of update_report submit and receipt messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of device that generated this message.
	ownerid	Owner ID provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.
Vehicle	-	Container of vehicle information. It contains multiple module elements.
	name	Name of the vehicle, if any.
	model	Type name of the device provided by the car manufacturer.
	modelid	Model name of the vehicle.

Element	Attribute in element	Description
	vehicleid	Vehicle id defined by a car manufacturer/supplier.
Device	-	Container of device information. It contains multiple module elements.
	name	Name of the device, if any.
	type	Type name of the device, such as "Power management ECU", "Seat belt control ECU", etc.
	model	Model name of the device.
	deviceid	Device id defined by a car manufacturer/supplier.
Module	-	Container of module information, which contains a hash element.
	moduleid	Module id is a unique id provided by a car manufacturer/a supplier.
	version	Version of this software module.
	nextversion	The version of the module update in progress, which is mainly used for sending a response message during an update.
	status	Result of the application of this module.
Hash	-	Hash is a container of a hash value and information of its hash algorithm.
	algorithm	Algorithm of the hash function (e.g., SHA-3, SHA-256, etc.).
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 25 – Example of update_report (submit) message

```
<message protocol="1.0" version="1.0.2" type="update_report"
subtype="submit" sessionid="{9AFFEB5A-F36B-4E05-819F-
5BDCD3A0E3EC}" ownerid="oid987239487" messageid="{3F7A6438-8306-
447E-A1BB-99CED4C2B6AD}" trustlevel="3">
    <Vehicle name="vehicleName" modelid="mid34987130" type="ECU"
model="modelName" vid="vid0987234"/>
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{1F6EDD6C-17D4-461B-8403-1E240E26464E}"
version="1.4.0" nextversion="" status="ok">
            <Hash algorithm="SHA-256">hash data here</ModuleHash>
        </Module>
        <Module moduleid="{4D168B58-26FA-4157-9703-A431D99C8438}"
version="2.4.34.0" nextversion="" status="ok">
            <Hash algorithm="SHA-256">hash data here</ModuleHash>
        </Module>
    </Device>
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{70628FDC-2282-4B2F-8A36-13445DED587A}"
version="3.5.45.0" nextversion=""
            <Hash algorithm="SHA-256">hash data here</ModuleHash>
        </Module>
    </Device>
    <IssuedTime "1903-07-01T00:00:00Z"/>
    <ExpirationTime "1903-07-01T00:00:00Z"/>
</message>
```

8.2.5.2 update_report (receipt) message

At the end of the sequence, the update server sends an update_report (receipt) message to VMG so that the vehicle recognizes the termination of the whole update procedure. The message format of update_report (receipt) is almost the same as diagnose (receipt) message.

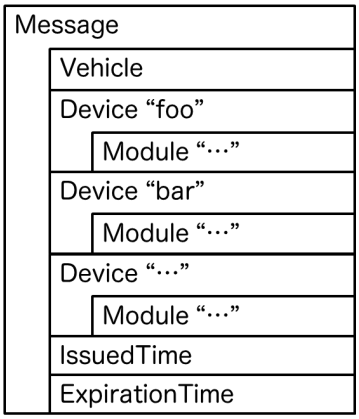

Figure 16 – Structure of update_report (receipt) message

Table 26 – Elements of update_report (receipt) message

Element	Attribute in element	Description
Message	-	Container of the message.
	protocol	Always "1.0".
	version	The version number of the message sender.
	type	Message type (always "update_report").
	subtype	Message subtype (always "receipt").
	sessionid	Session ID is a random GUID associated with the update_report session. An identical session ID is applied to a set of update_report submit and receipt messages.
	trustlevel	Trustlevel is determined based on the security capability and safety requirement of the device that generated this message.
	ownerid	Owner ID provided by a car manufacturer/supplier.
	messageid	Message ID is a random GUID associated with an individual message.
Vehicle	-	Container of vehicle information. It contains multiple module elements.
	name	Name of the vehicle, if any.
	model	Type name of the device provided by the car manufacturer.
	modelid	Model name of the vehicle.
	vehicleid	Vehicle id defined by a car manufacturer/supplier.
Device	-	Container of device information. It contains multiple module elements.
	name	Name of the device, if any.
	type	Type name of the device, such as "Power management ECU", "Seat belt control ECU", etc.
	model	Model name of the device.
	deviceid	Device id defined by a car manufacturer/supplier.
Module	-	Container of module information, which contains a hash element.
	moduleid	Module id is a unique id provided by a car manufacturer/a supplier.
	version	Version of this software module.
	nextversion	The version of the module update in progress, which is mainly used for sending a response message during an update.
	status	Acknowledgement of the report for this module.
IssuedTime	-	Time of generation of this message.
ExpirationTime	-	Expiration time of this message.

Table 27 – Example of update_report (receipt) message

```
<message protocol="1.0" version="1.0.2" type="update_report"
subtype="receipt" sessionid="{9AFFEB5A-F36B-4E05-819F-
5BDCD3A0E3EC}" ownerid="oid987239487" messageid="{B5585708-6BDA-
4B07-B2CB-5E9241F63271}" trustlevel="3">
    <Vehicle name="vehicleName" model="modelName"
modelid="mid34987130" vehicleid="vid0987234"/>
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{1F6EDD6C-17D4-461B-8403-1E240E26464E}"
version="" nextversion="1.3.23.0" status="ok"/>
        <Module moduleid="{4D168B58-26FA-4157-9703-A431D99C8438}"
version="2.4.34.0" nextversion="" status="ok"/>
    </Device>
    <Device name="device1" type="ECU" model="model1"
id="did0987234">
        <Module moduleid="{70628FDC-2282-4B2F-8A36-13445DED587A}"
version="3.5.45.0" nextversion="" status="ok"/>
        </Module>
    </Device>
    <IssuedTime "1903-07-01T00:00:00Z"/>
    <ExpirationTime "1903-07-01T00:00:00Z"/>
</message>
```

Appendix I

(This appendix does not form an integral part of this Recommendation.)

I.1 Methodology on risk analysis based on [b-JASO TP15002]

This appendix provides detailed information related to clause 7. This information is based on the guidelines concerning automotive information security [b-JASO TP15002].

Information security has become important in the design of the embedded system. Examples of the various security attacks are known for the IT system so far, and the know-how to evaluate a risk is accumulated by the IT systems design. The basic security concepts necessary for the evaluation of IT products are given in [ISO/IEC 15408-1]. In the context of evaluation, [ISO/IEC 15408-1] uses the term target of evaluation (TOE). There are assets which are entities that the owner of TOE presumably places value upon. [ISO/IEC 15408-1] aims to establish security objectives for a TOE, which is a statement of an intent to counter identified threats and/or satisfy identified organization security policies and/or assumptions. The threats give rise to risks to the assets, based on the likelihood of a threat being realized and the impact on the assets when that threat is realized. However, [ISO/IEC 15408-1] does not specify how to perform threat extraction and risk analysis.

Regarding the automotive security, the attack example of the automotive embedded system is poor at present. Therefore, it is difficult to carry out risk evaluation for the automotive embedded system design. Regarding the embedded system, this appendix describes the extracted threats and performs a risk analysis, according to the framework of [ISO/IEC 15408-1]. Here, it is aimed that the risk analysis does not depend on the know-how of the security design. Therefore, in this Recommendation, the risk analysis method CRSS [b-JASO TP15002] calculates a threat risk level of the embedded system. This method is characterized by the following: (1) We formulate the output in the system model definition step and the threat analysis step; (2) We set the value of the parameter using the information that was provided by carrying out the previous step.

The security evaluation process in [b-JASO TP15002] consists of the following phases:

Phase 1: Definition of the target of evaluation;

Phase 2: Identification of the threats;

Phase 3: Risk analysis.

Each phase is explained below.

I.1.1 Phase 1: Definition of the target of evaluation

The target for the identification of threats in the next phase is clarified.

The following four steps are performed in phase 1.

Step 1: Shared awareness establishment

Based on the system overview documentation, and in order for all the project members to establish a common awareness of life cycle of the target system, and the system construction, information such as systems construction figure, system function, and system-used data are prepared.

Step 2: Model figure construction of target of evaluation

A "model figure of target of evaluation" is constructed, which clarifies the system components and information flow between system components.

Step 3: Definition of an overview of the module functions

For each component module described in the model figure of the target of evaluation, the functions provided and their protected assets are clarified. In this way, a table of an "overview of module functions" is constructed.

Security threats can be described in terms of: "which threat agents exist, and what adverse action they perform on which assets" in the target system of evaluation. In addition to information, which has conventionally been treated as an asset to be protected, the assets in the automotive embedded systems also include embedded system software and the functions that control mechanisms such as the engine or brakes.

A system model is produced from the nature of the assets and from data flow diagrams that specify data flows in relation to these assets.

With respect to what adverse actions are performed (the threats), all of the things that can happen for each entry point are studied, and this is also considered in terms of the types of failure called confidentiality, integrity, or availability that can occur for each type of asset. For example, it is important that the functions of an automotive IT system operate as correctly as expected, and failure of integrity or availability must be prevented. Similarly, it is important that the information exchanged between central servers and vehicle-mounted intelligent transport system (ITS) devices is protected from disclosure and modification, and failure of confidentiality or integrity must also be prevented. Table I.1shows examples of information and other assets that the vehicles should protect.

Table I.1 – Examples of information and other assets that vehicles should protect (vehicle information security]

Objects that should be protected	Description
Operation of "Basic control functions"	Coherence and availability of "Basic control functions", execution environment of "Basic control functions", communications for the operation.
Information unique to the vehicle	Information which is unique to the body of the car (vehicle ID, device ID, etc.), authentication code, and accumulated information such as running history and operation history.
Vehicle status information	Data representing the vehicle's status such as location, running speed, and destination.
User information	Personal information, authentication information, billing information, usage history and operation history of the user (driver/passengers).
Software	Software which is related to vehicles' "Basic control functions" and "Expanded functions". Examples include software for electronic control unit (ECU).
Contents	Data for applications for video, music, map, etc.
Configuration information	Setting data for the behaviour of hardware, software, etc.

Step 4: Definition of target scope of a life cycle

A "table of life cycle" that clarifies the whole life cycles of the target system is constructed.

The threat agents are those people involved at any point along the life cycle of a vehicle, which includes its manufacturer, its use by the owners who purchased the vehicle be it new or second-hand, and ultimately its disposal. This is because confidential information held by the automotive embedded systems is stored and accessed not only during the normal usage phase but also at other

phases such as during the manufacturing, delivery, or servicing. The details of the TOE life cycle are shown in Table I.2.

Table I.2 – TOE life cycle

Phase	Sub-phase	Overview	Concerned persons
Operation	Transportation	An original equipment manufacturer (OEM) staff transports a manufactured vehicle to a vehicle dealer. He asks a transportation business operator to do this.	・OEM staff ・Transportation business operator ・Vehicle dealer staff ・Third party
	Vehicle delivery	A vehicle dealer staff delivers the vehicle to the owner.	・Vehicle dealer staff ・Owner ・Third party
	Normal operation/use	An owner or user uses the vehicle. A server administrator is involved as an administrator of updated server that provides software. A telecommunications carrier is involved to provide communication network.	・Owner or user ・Server administrator ・Telecommunications carrier ・Third party
	Normal operation/use Software download	To prepare for software update via the update server, the vehicle downloads the software from the update server.	・OEM staff ・Supplier staff ・Server administrator ・Telecommunications carrier ・Third party
	Maintenance (software update via update server) Software update	When the vehicle parks, software update is performed. The server administrator is involved as an administrator of the update server. The telecommunications carrier is involved as a communication network provider. A provider staff is involved as a service provider using the communication network.	・OEM staff ・Supplier staff ・Server administrator ・Telecommunications carrier ・Third party
	Maintenance (software update via OBD connector)	A vehicle dealer staff or a maintenance factory staff updates software via OBD connector at the time of vehicle inspection.	・Vehicle dealer staff ・Maintenance factory staff ・Owner or user ・Third party

I.1.2 Phase 2: Identification of threats

Security problems regarding TOE defined in phase 1 are identified.

The following three steps are performed in phase 2.

Step 1: Assumption setting

In order to clarify the scope where the threats are identified, assumptions are defined based on the model figure of the target of evaluation, the overview of module functions, and the table of the life cycle. The scope where the threats are identified in phase 2 is limited. The assumptions on the environment of TOE are defined. An identifier with a prefix "A" is assigned to each identified threat. In this way, a "table of assumptions" is constructed.

TOE is operated under the following assumptions:

A.Reliability_OfficeStaff (OEM staff/supplier staff/vehicle dealer staff /maintenance factory staff reliability)

OEM staff or supplier staff does not access physically the attack target vehicle. Furthermore, vehicle dealer staff/maintenance factory staff does not access physically the vehicle in normal operation/use phase.

A.Reliability_ServiceProvider (server administrator/telecommunications carrier reliability)

The update server administrator/telecommunications carrier does not access physically the vehicle. Furthermore, the update server administrator/telecommunications carrier does not intentionally cause threats.

A.Reliability_User (owner/user reliability)

An owner/user, in the maintenance phase, does not access physically the attack target vehicle.

An owner/user, in the maintenance phase, does not access physically the vehicle. Furthermore, the owner/user always locks the door in the normal operation/use phase. Furthermore, the owner/user does not allow concerned persons with whom he does not agree to access to in-vehicle in the normal operation/use phase.

A.Operation_Server (protection of server outside of target of evaluation)

The update server is properly operated, meaning that the concerned persons will not get/manipulate information stored in the server.

A.Control_OBD-Tool (protection of measuring device, etc., outside of target of evaluation)

A measuring device is properly operated, meaning that the concerned persons will not get/manipulate information stored in the measuring device.

Step 2: Identification of threats

Based on the model figure of the target of evaluation, the overview of module functions, and the table of the life cycle for each system component, the identity of the threats from the perspectives of where (entry points), who (threat agents), when (life cycle phase), why (reasons), and what (adverse actions) is shown in Table I.3. An identifier with a prefix "T" is assigned to each identified threat. In this way, a "table of threats" is constructed.

By applying these perspectives to the system model, the life cycle, and adverse actions, which are studied in defining the target system of the evaluation described in clause I.1, it can be exhaustively

identified which threat agents exist and which adverse actions they perform on which assets and at which phases.

Table I.3 – Perspectives for identification of threats

Perspective	Explanation
Where	Identify entry points of attacks.
Who	Identify threat agents.
When	Identify life cycle phases of attacks.
Why	Identify reasons for attacks.
What	Identify adverse actions.

Step 3: Organization's security policy setting

The organization's security policy defines the requirements that necessitate the security countermeasures due to reasons other than threats. Examples are laws and industry guidelines that need to be followed in the TOE development and in the operation environment. Laws or company rules concerning the system development that should be defined as security problems of the TOE are identified. An identifier with a prefix "O" is assigned to each security policy. In this way, a "table of the organization's security policy" is constructed.

There is no organization's security policy applied to TOE.

I.1.3 Phase 3: Risk analysis

This step specifies the degrees of risk for all the threats that have been identified.

For each threat in the table of threats, the degree of its priority is computed.

The following two steps are performed in phase 3.

Step 1: Risk evaluation

The risk that threats pose to an IT system has been typically assessed by deriving it from the value of the assets and the attack cost, which depends on how the threat is carried out. This is an effective approach when there are numerous examples of attacks, and a consensus can be reached about the cost of the attack method, including factors such as the execution time needed to undertake the attack and the capabilities of the person launching it. In the case of automotive embedded systems, while a number of example attacks have been identified at the research level, a wide range of attack method variations that exist for IT systems is not available. As a consequence, it is difficult to estimate the cost of attack methods.

I.1.3.1 CRSS

CVSS based Risk Scoring System (CRSS) is a threat risk assessment method, that is based on the common vulnerability scoring system (CVSS), i.e. the risk scoring system (RSS) of [ITU-T X.1521], used to score the severity of IT system vulnerabilities [b-JASO TP15002]. CVSS is composed of three metric groups: base, temporal, and environmental, each consisting of a set of metrics. These metric groups are described as follows:

- Base: represents the intrinsic and fundamental characteristics of a vulnerability that are constant over time and user environments.

- Temporal: represents the characteristics of a vulnerability that change over time but not among user environments.
- Environmental: represents the characteristics of a vulnerability that are relevant and unique to a particular user's environment.

CRSS evaluates risk scores by means of *base metric group* in CVSS scoring. The base metric group captures the characteristics of a vulnerability that are constant with time and across user environments. The access vector, access complexity, and authentication metrics capture how the vulnerability is accessed and whether or not extra conditions are required to exploit it.

The CRSS method assigns an asset value to each asset in terms of confidentiality, integrity, and availability and then it calculates a score for risk from the degree of ease in mounting an attack and from the degree of impact.

The degree of ease in mounting an attack is derived from the metric that reflects how close the threat agents need to get to the assets and from the existence of barriers that they break through to access them. A classification example with respect to the degree of ease in mounting an attack is shown in Table I.4.

The degree of impact measures how a vulnerability, if exploited, will directly affect an asset, where the impacts are independently defined as the degree of loss of confidentiality, integrity, and availability. For example, a vulnerability could cause a partial loss of integrity and availability, but no loss of confidentiality. A classification example with respect to the degree of impact is shown in Table I.5.

Table I.4 – Classification example with respect to degree of ease in mounting an attack (The format is referred to [b-JASO TP15002] Table D.2)

Parameter	Consideration principle	Classification	Examples
Access Vector (AC): Classification of the origin of the attack	Classification in terms of the origin (where) of the attack that caused the threat	Local (L)	USB memory
		Adjacent network (A)	Wi-Fi connection device
		Network (N)	Mobile line
Access Complexity (AC): The degree of the complexity of the attack condition	Classification in terms of the number of the required skills and knowledge of the attack	High (H)	Both skill and knowledge of the attack are required
		Medium (M)	Knowledge of the attack is required
		Low(L)	None (Little) skill and knowledge of the attack are required
Authentication (Au): The number of necessary authentication before attack	Classification in terms of the number of authentication between the asset and the threat agent	Multiple (M)	Multiple
		Single (S)	Single
		None (N)	Unnecessary

Table I.5 – Classification example with respect to degree of impact (The format is referred to [b-JASO TP15002] Table D.3)

Asset	Classification	C: Confidentiality impact			I: Integrity impact			A: Availability impact		
		None	Partial	Complete	None	Partial	Complete	None	Partial	Complete
Mobile communication function	Update service	Y					Y			Y
Mobile authentication information				Y			Y	Y		
Software get function		Y					Y			Y
Software				Y			Y	Y		
Remote software update function		Y					Y			Y
Software				Y			Y	Y		
GPS reception function	Information process	Y				Y			Y	
Wi-Fi connection function		Y				Y			Y	
Wi-Fi authentication information			Y			Y		Y		
USB connection function		Y				Y			Y	
CAN communication function	Vehicle control	Y					Y			Y
CAN gateway function		Y					Y			Y
Rooting table				Y			Y	Y		
OBD connection function		Y					Y			Y

For each threat described in the 5W (where, who, when, why, what) form, the risk score is computable.

Table I.6 – Risk scoring evaluation example (The format is referred to [b-JASO TP15002] Table D.4)

#	Threats	AV	AC	Au	Degree of ease of attack	C	I	A	Degree of impact	Risk value
1	T.control_fcn_Mobile_3rd_opearation_on_purpose of interfere-function	Network	Medium	Single		Un-necessary	Large	Large		
		1	0.61	0.56	6.83	0	0.66	0.66	9.20	7.95
2	T.vehicle_status_WiFi_dealer_main_purpose_forge	Adjacent network	Single(s)	Single		Small	Small	None		
		0.646	0.71	0.56	5.14	0.275	0.275	0	4.94	4.14

#	Threats	AV	AC	Au	Degree of ease of attack	C	I	A	Degree of impact	Risk value
3	T. info_transfer_USB_3rd_operation_pursuse_misop	Local	Low	None		None	Small	None		
		0.395	0.71	0.704	3.95	0	0.275	0	2.86	2.11

Even in cases such as the automotive embedded systems where there is a lack of accumulated know-how about security threats, the CRSS method can calculate a risk value analytically from the definitions of the threats and the system of evaluation. It can also incorporate consideration of factors such as risk to life into the risk assessment by treating functions as assets and, for the purpose of valuation, raising the estimated asset value in the case of functions for which loss of integrity or availability has serious consequences.

Step 2: Identifications of causes of threats

For each threat with a risk score of more than a certain value, the causes are logically analysed by means of a fault tree (FT).

I.2 Data verification using MAC algorithms

MAC algorithms play important roles in cryptography and security by achieving message integrity (authentication). In terms of MACs, ISO/IEC have produced significant results such as [b-ISO/IEC 9797-1] (Mechanisms using a block cipher), [b-ISO/IEC 9797-2] (Mechanisms using a dedicated hash-function), and [b-ISO/IEC 9797-3] (Mechanisms using a universal hash-function).

In order to select possible candidates for automotive security, MAC algorithms may have to give a clear security or performance advantage over the existing standardized MAC algorithms. The main target of MAC may achieve compact and fast software implementations on microcontrollers as well as to offer adequate security required by target applications. In particular, a very efficient MAC algorithm for microcontrollers can be desirable.

Bibliography

[b-ISO/IEC 9797-1] ISO/IEC 9797-1:2011, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher*.
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=50375

[b-ISO/IEC 9797-2] ISO/IEC 9797-2:2011, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanisms using a dedicated hash-function*.
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=51618

[b-ISO/IEC 9797-3] ISO/IEC 9797-3:2011, *Information technology – Security techniques – Message Authentication Codes (MACs) – Part 3: Mechanisms using a universal hash-function*.
http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=51619

[b-ISO/IEC 29192] Information technology -- Security techniques -- Lightweight cryptography
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=56425

[b-JASO TP15002] JASO TP15002:2015, *Guideline concerning automotive information security*.
[Editor's note] English version will be provided

[b-FIPS-202] Federal Information Processing Standards Publication-202 (2015), *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. National Institute of Standards and Technology,
http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

_____